ATTORNEY DOCKET NUMBER: 2006579-0141

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Laborczfalvi *et al.*	Examiner:	Morrison, Jay A.
Application No.:	10/711,737	Art Unit:	2168
Filed:	September 30, 2004	Conf. No.:	5736
For:	METHOD AND APPARATUS FOR ISOLATING EXECUTION OF SOFTWARE APPLICATIONS		

PRE-APPEAL BRIEF REQUEST FOR REVIEW

Mail Stop AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In response to the Final Office Action mailed August 22, 2007 and accompanying a Notice of Appeal to the Board of Appeals and Interferences in the United States Patent and Trademark Office appealing the final rejection of pending claims 1-32 in the above-referenced case, Applicants request a pre-appeal brief review and request consideration of the following remarks, pursuant to the July 12, 2005, Official Gazette Notice titled "New Pre-Appeal Brief Conference Pilot Program."

Applicants submit that (1) essential elements required to establish a prima facie rejection are omitted from the Examiner's rejection and (2) the Examiner's rejection contains clearly improper rejections based upon errors in facts.

The Examiner rejected claims 1-32 as anticipated by Czajkowski. Independent claim 1 recites:

> A method for isolating access by application programs to native resources provided by an operating system … redirecting to an isolation environment comprising a user isolation scope and an application isolation scope a request for a native resource made by a process executing on behalf of a first user… .

Independent claim 23 recites a similar limitation.

> An isolation environment for isolating access by application programs to native resources provided by an operating system, the isolation environment comprising … a redirector intercepting a request for the

native resource made by a process executing on behalf of the user and redirecting the request to the user isolation scope.

Czajkowski does not teach or suggest redirecting, to an isolation environment, a request for a native resource. Czajkowski merely describes a system for inspecting the source code in a Java program, or the Java byte code resulting from an intermediate compilation of a Java program, so that isolated applications may execute in a single virtual machine and may access shared classes. *See* Czajkowski, col. 11, lines 34-57. Czajkowski describes methods for separating out the static fields component of a class, maintaining a separate copy of static fields in shared classes, and creating a class including instance fields corresponding to one or more static fields. *See* Czajkowski, col. 11, lines 16-18 and 60-62; and col. 12, lines 40-45. Czajkowski describes creating a method class for each shared class, the method class including access methods for accessing fields extracted from shared classes. *See* Czajkowski, col. 12, lines 55-57. Describing a method for transforming the source code in a shared class into source code for three different classes for the purpose of enabling shared access to the class by object-oriented applications instantiating the classes does not teach a method for isolating access to a native resource provided by an operating system. Czajkowski himself provides the following description of classes used in object-oriented programming languages:

> In an object-oriented programming language, data and related methods can be grouped together or encapsulated to form an entity known as an object. All objects in an object-oriented programming system belong to a class, which can be thought of as a category of like objects which describes the characteristics of those objects. Each object is created as an instance of the class by a program. ... The class sets out variables and methods for objects which belong to that class. The definition of the class does not itself create any objects. The class may define initial values for its variables, and it normally defines the methods associated with the class ... The class may thereby provide all of the program code which will be used by objects in the class, hence maximizing re-use of code which is shared by objects in the class.

See Czajkowski, col. 8, lines 19-37. Applicants respectfully submit that this definition describes classes and objects that differ completely from the isolated resources described in the claimed invention.

In contrast to classes, on which Czajkowski is focused, the instant application describes resources provided by the operating system include, for example, a file system

and a registry database. *See* Specification, page 1. As described in the specification, file systems provide mechanisms for an application program to open, create, read, copy, modify, and delete data files, and registry databases provide functionality such as storing information regarding hardware physically attached to the computer, how computer memory is set up, various items of application-specific data, and what application programs should be present when the operating system is started. *See* Specification, page 1-2. One of ordinary skill in the art would not consider Java class members to be OS native resources. Czajkowski does not teach native resources provided by an operating system such as file systems or registry databases.

Czajkowski fails to address **inter-process** conflicts over operating system native resource. The pending claims are directed to applications that each have their own process, and addresses the problems that occur because the application processes share the same operating system instance. One of ordinary skill in the art would not find it obvious to solve the problem of inter-process OS native resource clashes based on a description of a solution for intra-process clashes. Even assuming for the sake of argument that native resources are opened using Java function calls to pass in a native resource name (a file path, a registry key name, the name of a named object such as an event or mutex, a COM object GUID, etc.), and that the name is the static field, several problems arise which are not addressed by Czajkowski. First, the names are strings, which are immutable in Java and hence would not need to be divided into per-application copies. Second, even giving each application its own copy of the name does not solve the inter-process conflicts, as each application would attempt to use the same native resource by passing the same name. Third, native resources in Windows can be accessed by traversing a path hierarchy from the root location down to the desired object. Czajkowski does not teach or suggest any mechanism to determine when two applications are using the same instance of a resource when they use different methods to navigate to the resource. For example, one application may open a file using path "C:\Directory1\Directory2\thefile.txt", while another application opens directory "C:\Directory1", and then navigates to subdir "Directory2", then opens the same version of the file - "thefile.txt".

Czajkowski addresses only clashes over system properties and, even then, teaches only that those clashes may be remedied by (1) letting all JAVA applications write to a

single, shared copy of the system property, (2) providing a read-only copy of each system property to every application, or (3) manually rewriting certain JAVA classes. First, JAVA system properties are not native resources, as that term is defined by the specification, for example, a file system or a registry database. *See* Specification, page 1. Rather, system properties are merely key-value maps maintained in memory by a particular JAVA Virtual Machine; they are not native resources. Further, any writes that are allowed to a particular system property are not persistent, they affect only the in-memory copy of the system property.

Accordingly, Applicants respectfully Czajkowski fails to teach each and every limitation of independent claims 1 and 23. Applicants respectfully request that the Examiner reconsider and withdraw the rejection of independent claims 1 and 23, and dependent claims 2-22 and 24-32.

CONCLUSION

In view of the above remarks, Applicants believe the pending application is in condition for allowance. Applicants believe no fee is due with this statement. However, if a fee is due, please charge our Deposit Account No. 03-1721.

Respectfully submitted,
Choate, Hall & Stewart, LLP

Dated November 21, 2007

By: /John D. Lanza/
John D. Lanza, Esq.
Registration No. 40,060
Attorney for Appellants

PATENT DEPARTMENT
CHOATE, HALL & STEWART, LLP
Two International Place
Boston, MA 02110
Tel: (617) 248-5000
Fax: (617) 248-4000